PN - JP7064766 A 19950310

PD - 1995-03-10

PR - JP19930208089 19930824

OPD - 1993-08-24

TI - MAXIMUM AND MINIMUM VALUE CALCULATING METHOD FOR PARALLEL COMPUTER

IN - SHINDO TATSUYA

PA - FUJITSU LTD

IC - G06F7/02 ; G06F15/16

PN - JP7064766 A 19950310

PD - 1995-03-10

AP - JP19930208089 19930824

IN - SHINDO TATSUYA

PA - FUJITSU LTD

TI - MAXIMUM AND MINIMUM VALUE CALCULATING METHOD FOR PARALLEL COMPUTER

AB - PURPOSE:To calculate maximum and minimum values in the parallel computer which consciously handles not only maximum and minimum values but also the value of an index to obtain the same result as a sequential computer.

- CONSTITUTION:Values of array elements as objects are compared with each other; and if they are equal to each other, index values are compared with each other, and the smaller index value is taken. That is, stages L1 to L9 where portions in charge of respective parts are individually calculated and a tournament processing is called and tournament processings (function global) L10 to L14 are equal to conventional procedure, but processings L29 to L30 executed in the case of equal inputs in functions to be subjected to maximum value comparison are different from conventional procedure. The minimum value is obtained in the same manner. Another way to take the larger index value in the case of equal inputs can be easily selected. This method is easily transformed that a parameter designates which of the larger index value and the smaller index value should be taken in the case of equal input values.

I - G06F7/02 ;G06F15/16

[Claim(s)]
[Claim 1] They are the maximum and the minimum value operation method characterized by comparing an index value when those values are equal, and taking an index value with the smaller value in the operation which calculates the maximum of an array element or the minimum value, and the index value that shows the position of the array element by the tournament method in a parallel computer while comparing the value of the target array element.
[Claim 2] The maximum and the minimum value operation method according to claim 1 characterized by comparing an index value and taking an index value with the larger value.
[Claim 3] The maximum and the minimum value operation method according to claim 1 or 2 characterized by comparing an index value and determining whether to take an index value with the larger value, or take the index value of the smaller one with the parameter defined beforehand.
[Claim 4] the case of a multidimensional array -- an index array element value -- REKISHIKO -- the claim 1 characterized by comparing with graphical order, or the maximum and the minimum value operation method according to claim 3

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平7－64766

(43)公開日　平成７年(1995) ３月10日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　7/02		M		
15/16	３９０	Ｔ　7429－5L		

審査請求　未請求　請求項の数４　ＯＬ　（全 6 頁）

(21)出願番号　特願平5－208089

(22)出願日　平成５年(1993) ８月24日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地

(72)発明者　進藤　達也
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(74)代理人　弁理士　井桁　貞一

(54)【発明の名称】　並列計算機における最大・最小値演算方法

(57)【要約】

【目的】並列計算機を用いた最大値・最小値を演算する方法に関し、逐次型と全く同じ結果を得ることができるようにすることを目的とする。

【構成】配列の要素の最大値または最小値と、その配列要素の位置を示すインデックス値とをトーナメント方式で求める演算において、対象とする配列要素の値を比較すると共に、それらの値が等しいときはインデックス値を比較して、その値の小さい方のインデックス値をとるようにする。

実施例のプログラム

```
        x = 0 ;                                                    L1
for ( j=担当分下限; j<= 担当分上限; j+= 刻み幅 ) {                  L2
    if ( a[j] (= x ) goto L100;                                    L3
    x = a[j];                                                      L4
    i = j;                                                         L5
L100:                                                              L6
    ;                                                              L7
}                                                                  L8
global(x, i, &x, &i);                                              L9

global(myData, myIndex, &myData, &myIndex) {                       L10
                                                                   ・
図4（B）に示す関数（トーナメント処理）                            ・
                                                                   ・
}                                                                  L14

最大値の比較（入力1，入力2，インデックス1，インデックス2，
              値の出力，インデックスの出力）{                      L20
    if（入力1＞入力2）{                                            L21
        値の出力＝入力1；                                          L22
        インデックスの出力＝インデックス1；                        L23
    } else if(入力1＜入力2）{                                      L24
        値の出力＝入力2；                                          L25
        インデックスの出力＝インデックス2；                        L26
    } else {                                                       L27
        値の出力＝入力1；                                          L28
        if（インデックス1＜インデックス2）{                        L29
        インデックスの出力＝インデックス1；                        L30
        } else {                                                   L31
        インデックスの出力＝インデックス2；                        L32
        }                                                          L33
    }                                                              L34
}                                                                  L35
```

1

【特許請求の範囲】

【請求項１】　並列計算機において、配列の要素の最大値または最小値と、その配列要素の位置を示すインデックス値とをトーナメント方式で求める演算において、対象とする配列要素の値を比較すると共に、それらの値が等しいときはインデックス値を比較して、その値の小さい方のインデックス値をとることを特徴とする最大・最小値演算方法。

【請求項２】　インデックス値を比較して、その値の大きい方のインデックス値をとることを特徴とする請求項１に記載の最大・最小値演算方法。

【請求項３】　インデックス値を比較して、その値の大きい方のインデックス値をとるか小さい方のインデックス値をとるかを、あらかじめ定めたパラメータにより決定することを特徴とする請求項１または請求項２に記載の最大・最小値演算方法。

【請求項４】　多次元配列の場合にインデックス配列の要素値をレキシコグラフィカルな順に比較することを特徴とする請求項１ないし請求項３に記載の最大・最小値演算方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】数値計算の中で、配列要素の最大値または最小値とその要素の位置を示すインデックスを求める演算がよく使われる。

【０００２】高速演算用の計算機の一つとして並列計算機がある。本発明は、並列計算機を用いた最大値・最小値を演算する方法に関する。

【０００３】

【従来の技術】並列計算機を用いて最大値または最小値を高速に求める方法として、トーナメント方式が知られている。並列計算機では、対象となる数値群（配列の要素）を各プロセサに配分し、それぞれ最大値（最小値）を求め、そのあと各プロセサのもつ最大値（最小値）の間の最大値（最小値）を求める。各プロセサがそれぞれ配分された値の最大値（最小値）を求める段階は当然に並列に処理が行われる。その後の各プロセサにある最大値（最小値）の間の全体の最大値（最小値）を求める段階でもできるだけ複数のプロセサを並列的に動作させるためトーナメント方式をとる。

【０００４】図３は並列計算機の構成説明図である。プロセサエレメントPEを並列に多数並べ、その間をネットワークｎで結合したものである。図は８×８のプロセサエレメント（以下プロセサと記す）を２次元トーラスネットワークで結合した例である。図４はこのような並列計算機において行なうトーナメント方式の演算の概念を示す図である。各プロセサ（そのもつ値）を２つずつ組にして、１つのプロセサで、１つの値を他のプロセサから転送してもらい、それらのより大なる値（またはより小なる値）を求める。その結果を再び２つずつ組にして

2

・・という演算を繰り返し、最終的に１つの値を解として、すなわち最大値（または最小値）として得る。図４（Ａ）では、１回目にはプロセサ０と１、２と３、４と５、６と７、・・・が組になり、プロセサ１，３，５，７，・・・がプロセサ０，２，４，６，・・・にデータを送り、そこでそれぞれ演算を行なう（①で示す）。次にプロセサ０と２、４と６、・・・が組になり、プロセサ２，６・・・がプロセサ０，４・・・にデータを送り、そこでそれぞれ演算を行なう（②で示す）。次にプロセサ０と４、・・・が組になり・・・（③で示す）ということを繰り返し、最後にプロセサ０に最終結果としての最大値が得られる。

【０００５】図４（Ｂ）に以上のトーナメント処理を疑似Ｃ言語で関数として記述したプログラム例を示す。この例では、プロセサの数が2^3個であり、変数pid にそれぞれプロセサ固有の番号（機番）が設定されているものとする。各プロセサはトーナメント処理の対象となる値とそのインデックスとをmyDataとmyIndex に設定しておき、それぞれこのプログラムを実行すると最終的にpid=0 番のプロセサのmyDataとmyIndex に結果が得られる。なお、sendはメッセージを転送する関数であり、第１引数で示されるプロセサに第２と第３の引数を送り、recvはメッセージを受信する関数であり、sendで自分に送られてきたメッセージを受ける。演算関数は、「最大値の比較」や「最小値の比較」等の演算の関数を呼び出す部分である。

【０００６】最大値演算等を行なうとき、プロセサは、その要素の配列上の位置を示すインデックスを要素に付随させて扱う。つまり、２組の要素を比較し、結果の値と共にそのインデックス値を付随させる。従って、比較の結果残った方（勝った方）の値と共に付随するインデックス値も勝ち残り、最終的に残った最大値（または最小値）とそれに付随するインデックス値が解として得られる。これにより、その最大値（または最小値）が配列のどの位置であるか、またそれをもっていたプロセサがどれかが分かり、その後の演算のための指標となる。

【０００７】ところで、比較する２つの値が等しい場合にインデックス値としてどちらの値をとるかはあまり意識されていない。最大値または最小値が正しければインデックス値そのものは２義的なものであるからである。

【０００８】図２に逐次型最大値プログラムと従来の最大値計算プログラムとを示す。（Ａ）の逐次型プログラムは、配列ａの要素を１からＮまで調べて、最も大きな値をｘに、そのときのインデックス値をｉに結果として得るプログラムで、逐次計算機用に書かれたものである。

【０００９】（Ｂ）は同様の処理を並列計算機用に書いたものである。まず、各プロセサ内で担当するａの要素について最大値とそのインデックスを求める（L1～L8）。次にトーナメント処理をおこなう関数global（L

9) を呼出し、各プロセサが計算した最大値の中でさらに最大のものとそのインデックスとを求める。ここで、関数globalは図４（B）に示したものであり、第１引数ｘは比較対象のデータを与える変数、第２引数ｉはそのインデックス値を与える変数、第３引数＆ｘは各プロセサが設定したｘの最大値（最小値）を結果として返す変数、第４引数＆ｉはそれに付随するインデックスとして返す変数である。これを全てのプロセサが実行する。

【００１０】逐次型も並列型も結果として得られるｘの値は常におなじである。しかし、そのときのインデックスの値ｉは異なる場合がある。各プロセサが関数globalの第一引数として設定するｘが同じものが複数存在する場合である。逐次型のプログラムでは図２（A）のＬ３に示す条件分岐の性質により、インデックス値の最も小さいものが最終結果として採用される。一方、並列型のプログラムではプロセサ間の大小比較をトーナメント方式による並列処理によって行っているため、プロセサへの要素の割り付け方により、最も小さなインデックス値のものが採用されることは保証されない。その結果、もともとは逐次処理用に書かれたプログラムを、このような方法で並列計算機用に直した場合に、異なる結果が得られる場合がある。また並列化したプログラムを異なるプロセサ数の構成で実行した場合に、各プロセサに割りつけられるデータが変化するため、結果が異なる場合があり得る。

【００１１】

【発明が解決しようとする課題】通常の逐次計算機用に開発されたプログラムを並列計算機用に並列化した場合に、逐次計算機で得た結果とは、インデックス値が異なることがある。また、並列計算機においてプロセサの数が異なると、結果のインデックス値が異なることがある。それにより最終的な結果が微妙に異なることがある。このような状況は、プログラムが正しく動作しているか、あるいは正しく並列化されたかの判定を困難にする。すなわち、結果の互換性の問題がある。

【００１２】本発明は、最大値／最小値だけでなく、インデックスの値も意識して取り扱うことにより、逐次型と全く同じ結果を得ることができるようにした、並列計算機での最大・最小値演算方法を実現することを目的としている。

【００１３】

【課題を解決するための手段】図１は本発明の実施例のプログラムを示すものである。図の L20～L35 に示す最大値の比較を行なう関数の L29～L32 のようにインデックス値の大小関係を意識した処理を行なう。

【００１４】請求項１に記載した発明は、配列の要素の最大値または最小値と、その配列要素の位置を示すインデックス値とをトーナメント方式で求める演算において、対象とする配列要素の値を比較すると共に、それらの値が等しいときはインデックス値を比較して、その値

の小さい方のインデックス値をとるようにする。

【００１５】請求項２に記載したものは、前記の演算において、インデックス値を比較して、その値の大きい方のインデックス値をとるようにした場合である。請求項３に記載したものは、前２項の演算において、インデックス値を比較して、その値の大きい方のインデックス値をとるか小さい方のインデックス値をとるかを、あらかじめ定めたパラメータにより決定するようにした場合である。

【００１６】演算対象が多次元配列の場合には、インデックスはそれぞれ、演算対象の多次元配列の次元数の要素数をもった１次元配列となる。請求項４に記載したように、演算対象の比較値が等しい場合に、それらのインデックス配列の要素値をレクシコグラフィカル（辞書順ともいう）に比較することで解決する。

【００１７】

【作用】請求項１又は２に記載したようにすることにより、並列計算機のプロセサの台数が変わっても結果は常に同じであることが保証できる。

【００１８】請求項３に記載したようにすることにより、逐次プログラムを並列化する場合に、もとのループの判定条件によりインデックス値のとりかたを使い分けることができる。例えば図２（A）の例では判定条件が

if（ a[j] <= x ）goto L100;

であるので，インデックス値の小さい方をとるように設定すればよい。仮に判定条件が、

if（ a[j] < x ）goto L100;

になっていれば、インデックス値の大きい方をとるように設定すればよい。以上の例でもしループが降順になっているなら、それぞれ逆にインデックス値の大きい方、小さい方をとるように設定することで対応できる。

【００１９】多次元配列を演算対象とする場合も、インデックスが１次元配列となるのでその比較をレクシコグラフィカルに行なう点を除けば、同様である。このようにすることにより、最大値・最小値演算を並列計算機で演算する場合に、インデックス値を一意に定めることができる。従って、プロセサの数によらない結果の互換性、逐次型との結果の互換性を保証することができる。

【００２０】

【実施例】以下、図面を参照して本発明の実施例を説明する。図１は本発明の一実施例として最大値を求めるサブルーチンを示すものである。入力の値が等しい場合はインデックス値の小さい方をとる例である。

【００２１】L1 ～L9の、個別に担当分を演算し、トーナメント処理を呼ぶまでの段階と、L10～L14 のトーナメント処理（関数global）そのものは従来と同じである。最大値の比較を行なう関数内のＬ29～Ｌ32に示す、入力が等しい場合の処理が本発明の方法の要部である。

【００２２】最小値を求める場合も同様にすればよいことはいうまでもない。入力の値が等しい場合はインデッ

5

クス値の大きい方をとるようにすることも容易にできる。また、入力の値が等しい場合はインデックス値の大きい方をとるか小さい法をとるかパラメータにより指定するようにする変形も容易であることはいうまでもない。

【００２３】演算対象の配列は１次元に限らず多次元であってもよい。その場合はインデックスも配列となるので、インデックスの配列同士をレキシコグラフイカルな順に比較して大小関係を判断する。

【００２４】なお、配列の要素は数値である必要はなく、文字列であってもよい。この場合もレキシコグラフイカルな順に比較して大小関係を判断すればよい。

【００２５】

【発明の効果】以上説明したように、本発明によれば最大値・最小値を求める演算を並列化した場合に逐次型プログラムとの解の互換性を保証することができる。また、並列化したプログラムを実行する場合に、プロセサの数によらず常に解が変わらないことを保証できる。このような性質は、並列化したプログラムがもとのプログラムの意味を損なわず正しく変換されたことを保証する上で重要である。

6

【図面の簡単な説明】

【図１】本発明の一実施例のプログラムの要部を示す図である。

【図２】従来の最大値演算プログラムの例を示す図である。

【図３】並列計算機の構成説明図である。

【図４】トーナメント処理の説明図である。

【符号の説明】

PE　プロセサ（エレメント）　　n　プロセサ間ネットワーク

【図１】

実施例のプログラム

```
        x＝０；                                          L1
for ( j=担当分下限; j<= 担当分上限; j+= 刻み幅 ) {      L2
    if ( a[j] <= x ) goto L100;                          L3
    x = a[j];                                            L4
    i = j;                                               L5
L100:                                                    L6
    ;                                                    L7
}                                                        L8
global(x, i, &x, &i);                                    L9

global(myData, myIndex, &myData, &myIndex)  {            L10
                                                         ・
図４（Ｂ）に示す関数（トーナメント処理）                 ・
                                                         ・
}                                                        L14


最大値の比較（入力１，入力２，インデックス１，インデックス２，
              値の出力，インデックスの出力）  {          L20
    if（入力１＞入力２）  {                              L21
        値の出力＝入力１；                               L22
        インデックスの出力＝インデックス１；             L23
    } else if(入力１＜入力２)  {                         L24
        値の出力＝入力２；                               L25
        インデックスの出力＝インデックス２；             L26
    } else {                                             L27
        値の出力＝入力１；                               L28
        if（インデックス１＜インデックス２）  {          L29
        インデックスの出力＝インデックス１；             L30
        } else {                                         L31
        インデックスの出力＝インデックス２；             L32
        }                                                L33
    }                                                    L34
}                                                        L35
```

【図２】

従来の最大値演算プログラムの例

（Ａ）逐次型プログラム

```
        x＝０；                                          L1
for ( j=1; j<N; j++) {                                   L2
    if ( a[j] <= x ) goto L100;                          L3
    x = a[j];                                            L4
    i = j;                                               L5
L100:                                                    L6
    ;                                                    L7
}                                                        L8
```

注：　a[j]　配列　（入力）
　　x　　最大値
　　i　　そのインデックス

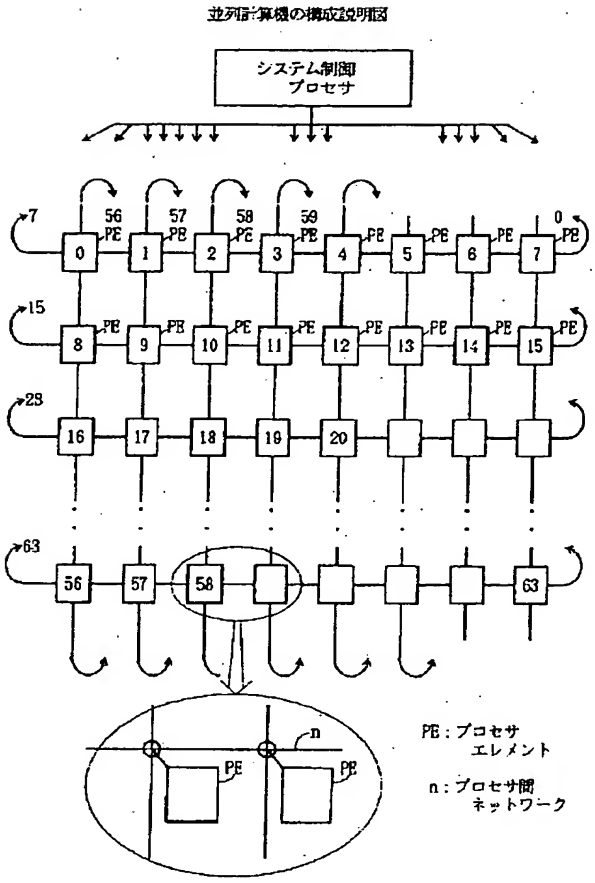
（Ｂ）並列型プログラム

```
        x＝０；                                          L1
for ( j=担当分下限; j<= 担当分上限; j+= 刻み幅 ) {      L2
    if ( a[j] <= x ) goto L100;                          L3
    x = a[j];                                            L4
    i = j;                                               L5
L100:                                                    L6
    ;                                                    L7
}                                                        L8
global(x, i, &x, &i);                                    L9

global(myData, myIndex, &myData, &myIndex)  {            L10
                                                         ・
図４（Ｂ）に示す関数（トーナメント処理）                 ・
                                                         ・
}                                                        L14


最大値の比較（入力１，入力２，インデックス１，インデックス２，
              値の出力，インデックスの出力）  {          L20
    if( 入力１＞入力２）  {                              L21
        値の出力＝入力１；                               L22
        インデックスの出力＝インデックス１；             L23
    } else {                                             L24
        値の出力＝入力２；                               L25
        インデックスの出力＝インデックス２；             L26
    }                                                    L27
}                                                        L28
```

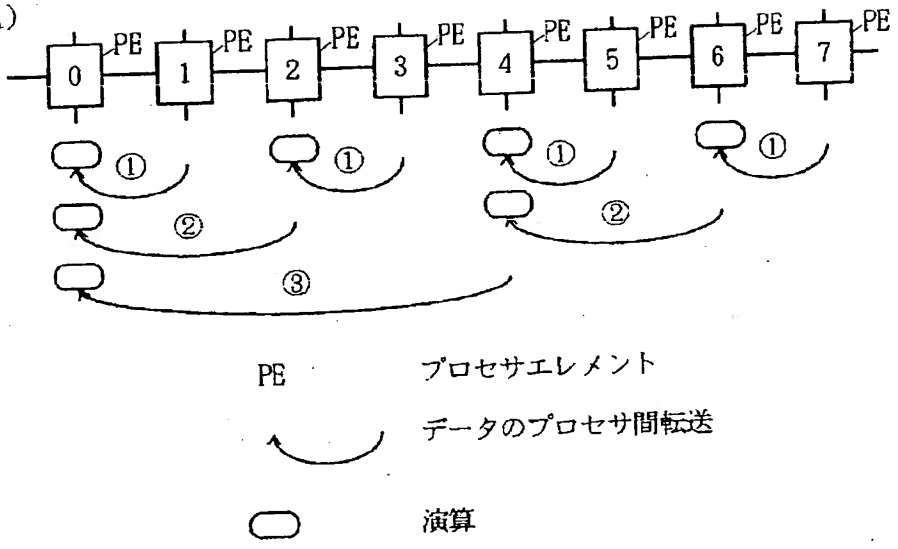
【図３】



【図４】

トーナメント処理の説明図

(A)



(B) 上図の処理を疑似C言語の関数として記述したプログラム例

```
global(myData, myIndex, &myData, &myIndex)  {
  sendFlag=0;
  for(i=0; i<n; i++) {
    if(sendFlag==0)  {
      if(pidのi ビット目==1) {
        send(pidの iビット目を 0に置き換えたプロセサID, myData, myIndex);
        sendFlag=1;
      } else { /*  pidの iビット目==0 */
        recv(&rcvData, &rcvIndex);
        演算関数(myData, rcvData, myIndex, rcvIndex, &outData, &outIndex);
        myData=outData;
        myInex=outIndex;
      }
    }
  }
}
```

注：
1. プロセサ数は 2^n 個、pid は各プロセサの機番を設定する変数
2. myData, myIndexは演算対象となるデータとインデックス
3. sendは第１引数のプロセサに第２第３引数の値を送る関数
4. recvはsendで送られてきた値を受け取る関数
5. 演算関数は最大値・最小値比較等の演算を行なう関数
6. 最終結果はpid=0 のプロセサのmyData, myIndexに残る。

PN - SU1546960 A 19900228

PD - 1990-02-28

PR - SU19884440848 19880614

OPD - 1988-06-14

TI - DEVICE FOR DETERMINING EXTREME VALUES

IN - VASILKEVICH ALEKSANDR V (SU); DMITRIEV ALEKSANDR G (SU); ELMANOV SERGEJ A (SU); MIKHAJLOVICH IGOR V (SU); VASILENKO LARISA I (SU)

PA - VASILKEVICH ALEKSANDR V (SU); DMITRIEV ALEKSANDR G (SU); ELMANOV SERGEJ A (SU); MIKHAJLOVICH IGOR V (SU); VASILENKO LARISA (SU)

IC - G06F7/02

TI - Two-dimensional data files extremal values finder - has pairs of comparators processing fragment line values to select max. and min. for memories

PR - SU19884440848 19880614

PN - SU1546960 A 19900228 DW199038 000pp

PA - (VASI-I) VASILKEVICH A V

IC - G06F7/02

IN - DMITRIEV A G; ELMANOV S A; VASILKEVIC A V

AB - SU1546960 Appts. comprises memories (1,2), subtracting counters (3-6), registers (7-14), comparators (15-18), commutators (19,20), AND-gates (21,22), delay elements (23,24), univibrator (25), clock input (26) and data input (27).

- The two-dimensional matrix or frame is split into L x H rectilinear fragments each having dims. of N x M elements. The appts. determines the max. and min. elements from each of the fragments, memories (1,2) having a capacity of L x H cells. The frame values pass sequentially to input (27) and comparators (17,18) find the max. and min. values of the centre of the values of one line before processing the line of for the next fragment.
- USE/ADVANTAGE - Appts. is for use in high-speed specialised computers in processing two-dimensional data e.g. TV images in real time. Appts. can now determine the extrema of two-dimensional signals by rectilinear fragments. Bul. 8/28.2.90 (3pp Dwg.No. 1/1)

OPD - 1988-06-14

AN - 1990-288983 [38]